

This Page Is Inserted by IFW Operations  
and is not a part of the Official Record

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning documents *will not* correct images,  
please do not report the images to the  
Image Problem Mailbox.**



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/743,333	12/22/2003	Laurent Ugen	S1022.81016US01	5580

23628 7590 07/13/2004

WOLF GREENFIELD & SACKS, PC  
FEDERAL RESERVE PLAZA  
600 ATLANTIC AVENUE  
BOSTON, MA 02210-2211

EXAMINER
----------

HUISMAN, DAVID J

ART UNIT	PAPER NUMBER
----------	--------------

2183

DATE MAILED: 07/13/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

**Office Action Summary**

Application No.

10/743,333

Applicant(s)

UGEN ET AL.

Examiner

David J. Huisman

Art Unit

2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1) ☒ Responsive to communication(s) filed on 22 December 2003.
- 2a) ☐ This action is FINAL. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4) ☒ Claim(s) 1-31 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-31 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

**Application Papers**

- 9) ☒ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 14 April 2004 is/are: a) ☐ accepted or b) ☒ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

- 12) ☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
  2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)  
Paper No(s)/Mail Date 4/14/2004
- 4) ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date. \_\_\_\_\_
- 5) ☐ Notice of Informal Patent Application (PTO-152)
- 6) ☐ Other: \_\_\_\_\_

**DETAILED ACTION**

1. Claims 1-31 have been examined.

***Papers Submitted***

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: IDS, Drawings, and Declaration as received on 4/14/2004.

***Information Disclosure Statement***

3. Applicant has submitted a list of copending applications which includes application # 09/563,186. A copy of this application was not provided and therefore, this particular application has not been considered.

***Priority***

4. On page 1, lines 5-11 of the specification, it is stated that U.S. Patent Application 09/563,703 has a filing date of February 27, 2001. However, this application actually has a filing date of May 2, 2000. Applicant is asked to correct this error.
5. Applicant is reminded that in order for a patent issuing on the instant application to obtain the benefit of priority based on priority papers filed in parent Application No. 09/563,703 under 35 U.S.C. 119(a)-(d) or (f), a claim for such foreign priority must be made in this application. In making such claim, applicant may simply identify the application containing the priority papers.

***Oath/Declaration***

6. The oath or declaration is defective. A new oath or declaration in compliance with 37 CFR 1.67(a) identifying this application by application number and filing date is required. See MPEP §§ 602.01 and 602.02.

The oath or declaration is defective because the first inventor's last name is spelled incorrectly. That is, from a disclosed foreign application (EP 1,050,805 A1) and other U.S. Patents issued to the same inventor, the examiner is under the impression that the first inventor's last name should be spelled "Uguen" and not "Ugen".

***Specification***

7. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

8. The lengthy specification has not been checked to the extent necessary to determine the presence of all possible minor errors. Applicant's cooperation is requested in correcting any errors of which applicant may become aware in the specification.

9. The disclosure is objected to because of the following informalities: On page 9, if line 33 is the beginning of a new paragraph, please indent the line. Otherwise, remove the space between lines 32 and 33. On page 22, if line 25 is the beginning of a new paragraph, please indent the line. Otherwise, remove the space between lines 24 and 25. On page 25, line 29, replace reference number "212" with --221--. Replace the first occurrence of "as" on page 17, line 28, with --at--. Finally, on page 3, line 17, and on page 3, line 32, the use of the word "which" is improper. Please reword the phrases such that they read properly and clearly.

Appropriate correction is required.

### *Drawings*

10. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they do not include the following reference sign(s) mentioned in the description: Reference numbers 210, 212, 214, and 216 are not shown in Fig.10. However, they are mentioned in the description on page 25, lines 4-5. A proposed drawing correction or corrected drawings are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

11. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they include the following reference sign(s) not mentioned in the description: Reference number 120, shown in Fig.1, is not mentioned within the description. A proposed drawing correction, corrected drawings, or amendment to the specification to add the reference sign(s) in the description, are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

### *Double Patenting*

12. A rejection based on double patenting of the "same invention" type finds its support in the language of 35 U.S.C. 101 which states that "whoever invents or discovers any new and useful process ... may obtain a patent therefor ..." (Emphasis added). Thus, the term "same invention," in this context, means an invention drawn to identical subject matter. See *Miller v. Eagle Mfg. Co.*, 151 U.S. 186 (1894); *In re Ockert*, 245 F.2d 467, 114 USPQ 330 (CCPA 1957); and *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970).

A statutory type (35 U.S.C. 101) double patenting rejection can be overcome by canceling or amending the conflicting claims so they are no longer coextensive in scope. The

Art Unit: 2183

filing of a terminal disclaimer cannot overcome a double patenting rejection based upon 35 U.S.C. 101.

13. Applicant is advised that should claim 15 be found allowable, claim 19 will be objected to under 37 CFR 1.75 as being a substantial duplicate thereof. When two claims in an application are duplicates or else are so close in content that they both cover the same thing, despite a slight difference in wording, it is proper after allowing one claim to object to the other as being a substantial duplicate of the allowed claim. See MPEP § 706.03(k).

### *Claim Objections*

14. Claim 1 is objected to because of the following informalities: In line 13 (page 31), replace "said one execution unit" with --said at least one of the plurality of execution units--.

15. Claim 15 is objected to because of the following informalities: In line 8 on page 34, replace "send guard" with --sendguard--. Appropriate correction is required.

16. Claim 19 is objected to because of the following informalities: In line 20 on page 35, replace "send guard" with --sendguard--. Appropriate correction is required.

17. Claim 20 is objected to because of the following informalities: In line 29 on page 35, replace "in which" with something more grammatically correct. In lines 28, 32, and 33 on page 35, insert --processing-- before "unit". In line 1 on page 36, replace "the first one" with --the first FIFO-type memory--. In lines 2, 4, 5, and 9 on page 36, replace "first memory" with --first FIFO-type memory--. In line 3 on page 36, insert --processing-- before "unit". In lines 3, 7, and 10 on page 36, replace "second memory" with --second FIFO-type memory--. Appropriate correction is required.

Art Unit: 2183

18. Claim 21 is objected to because of the following informalities: Insert a comma after "memory" in line 13 on page 36. Replace all occurrences of "second memory" with --second FIFO-type memory--. Finally, the last paragraph of claim 21 is confusing and should be reworded so that it is more clear. More specifically, the examiner finds that the phrase "in that determining" confusing. Also, the examiner recommends replacing "account of" with --into account-- in line 20 on page 36. Appropriate correction is required.

19. Claim 22 is objected to because of the following informalities: Replace all occurrences of "second memory" with --second FIFO-type memory--. Appropriate correction is required.

20. Claim 23 is objected to because of the following informalities: Replace all occurrences of "first memory" with --first FIFO-type memory--. Replace all occurrences of "second memory" with --second FIFO-type memory--. Insert a comma after "stored" in line 7 on page 37. Also, either remove the comma after "instruction" in line 8 on page 37, or insert --the label-- before "containing". Finally, in line 13 on page 37, replace "this" with either --said-- or --the--. Appropriate correction is required.

21. Claim 24 is objected to because of the following informalities: Replace all occurrences of "first memory" with --first FIFO-type memory--. Replace all occurrences of "second memory" with --second FIFO-type memory--. Replace all occurrences of "this" with either --said-- or --the--. Finally, the language of the first paragraph of claim 24 is very confusing to the examiner. The examiner recommends rewording this claim so that it reads more clearly upon allowance. Appropriate correction is required.

22. Claim 25 is objected to because of the following informalities: Replace all occurrences of "second unit" with --second processing unit--. In line 1 on page 38, replace "the first one"

with --the first FIFO-type memory--. Replace all occurrences of "first memory" with --first FIFO-type memory--. Replace all occurrences of "second memory" with --second FIFO-type memory--. Appropriate correction is required.

23. Claim 26 is objected to because of the following informalities: Replace all occurrences of "first memory" with --first FIFO-type memory--. Replace all occurrences of "second memory" with --second FIFO-type memory--. Appropriate correction is required.

24. Claim 27 is objected to because of the following informalities: Replace all occurrences of "second memory" with --second FIFO-type memory--. Appropriate correction is required.

25. Claim 29 is objected to because of the following informalities: Replace all occurrences of "first memory" with --first FIFO-type memory--. Appropriate correction is required.

26. Claim 30 is objected to because of the following informalities: Replace all occurrences of "first memory" with --first FIFO-type memory--. Appropriate correction is required.

### *Claim Rejections - 35 USC § 112*

27. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

28. Claims 2-5 are rejected because claim 2 recites the limitation "the execution unit" in line 20 on page 31. There is insufficient antecedent basis for this limitation in the claim.

29. Claims 9-10 are rejected because claim 9 recites the limitation "said execution unit" in line 25 on page 32. There is insufficient antecedent basis for this limitation in the claim.

30. Claim 20 recites the limitation "the respective guarded instruction" in line 27 on page 35. There is insufficient antecedent basis for this limitation in the claim.

31. Claim 21 recites the limitation "the current value of the read counter" in line 21 on page 36. There is insufficient antecedent basis for this limitation in the claim.

32. Claims 22 and 24 are rejected because claim 22 recites the limitation "the current value of the overflow bit" in line 29 on page 36. There is insufficient antecedent basis for this limitation in the claim.

33. Claim 23 recites the limitation "the memory-stored current value" in line 13 on page 37. There is insufficient antecedent basis for this limitation in the claim.

34. Claim 25 recites the limitation "the first processing unit" in line 30 on page 37. There is insufficient antecedent basis for this limitation in the claim. Please change all occurrences of "first unit" to --first processing unit--.

35. Claim 26 recites the limitations "the set of guard indications" in lines 18-19 on page 38, "the current value of the write counter" in line 21 on page 38, and "the current value of the read counter" in lines 25-26 on page 38. There is insufficient antecedent basis for these limitations in the claim.

36. Claims 27-28 are rejected because claim 27 recites the limitation "the current value of the overflow bit" in lines 5-6 on page 39. There is insufficient antecedent basis for this limitation in the claim.

37. Claim 29 recites the limitations "the corresponding guard indication" in lines 16-17 on page 39, "the input stage" in line 17 on page 39, and "the third memory" in line 19 on page 39. There is insufficient antecedent basis for these limitations in the claim.

38. Claims 30-31 are rejected because claim 30 recites the limitations "the auxiliary field" in lines 26-27 on page 39, "the content of the second supplementary field" in line 30 on page 39,

and "the current value of the overflow bit" in lines 30-31 on page 39. There is insufficient antecedent basis for these limitations in the claim.

***Claim Rejections - 35 USC § 103***

39. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

40. Claims 1-14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Witt et al., U.S. Patent No. 5,651,125 (herein referred to as Witt) in view of Intel, "IA-64 Application Developer's Architecture Guide" (herein referred to as Intel).

41. Referring to claim 1, Witt has taught a computer system for executing instructions, the system comprising:

a) instruction supply circuitry and a plurality of parallel execution units for receiving respective instructions from the supply circuitry. See Fig. 1, and note that instructions are supplied to parallel execution units 90, 95, 100, 105, 60, and 65.

b) Witt has not taught that each instruction has a respective guard indicator selected from a set of guard indicators, common to the plurality of execution units. However, Intel has taught the idea of instructions using a predicate selected from a set of 64 predicate registers. See section 3.1.4 on page 3-4 and also note the instructions on pages 7-8, 7-19, 7-41, and 7-127. Each instruction references (qp) which is a qualifying predicate. The value of this predicate will determine whether or not or how the instruction will execute. By employing such a feature, conditional

execution of all instructions may be realized. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Witt so that Witt's instructions include respective guard indicators (predicates).

c) one of said execution units including a master guard value store containing a master representation of current values for the guard indicators in said set of guard indicators and guard value transfer circuitry operable to transfer a guard value from said master store to another of said execution units in response to a sendguard instruction being executed in said one execution unit. When a guarded (predicated) branch instruction is executed in Witt in view of Intel, the guard value (predicate) will need to be sent from the predicate register file to the branch execution unit so that it can be evaluated. In this sense, the branch instruction, or any other instruction which is guarded, is a "sendguard" instruction. Furthermore, as taught by In re Japikse 86 USPQ 70 (CCPA 1950), to shift location of parts is generally not given patentable weight or would have been an obvious improvement. More specifically, the master predicate register file may be located anywhere within Witt because no matter where it is located, a guard value must still be sent to the appropriate place for evaluation.

d) the instruction supply circuitry comprises a main instruction queue for holding instructions to be supplied to the parallel execution units and a subsidiary instruction queue for holding sendguard instructions, with the subsidiary queue having priority access to the execution pipelines to avoid unnecessary delays for execution of sendguard instructions. See Fig. 1A and note that each parallel unit has a queue 90R, 95R, 100R, etc. The branch execution unit queue is considered a subsidiary queue because it holds branch instructions for branch execution and branch instructions have high priority. See column 25, lines 12-15. Therefore, branch

instructions (sendguard instructions) would not cause delay in the pipeline because they have priority. It should be realized from Fig. 1A that the parallel units each have their own queue. However, a person of ordinary skill in the art would have recognized that any of the queues are combinable to form a main instruction queue and the functionality would still be the same, and that such a combination would be based on user preference. In re Larson 144 USPA 347 (CCPA 1965), has taught that to make integral is generally not given patentable weight or would have been an obvious improvement. That is, it would have been obvious to one of ordinary skill in the art at the time of the invention to integrate every queue of Witt, except for the subsidiary queue, to form a single main instruction queue.

42. Referring to claim 2, Witt in view of Intel has taught a computer system as described in claim 1. Intel has further taught that said execution unit includes circuitry for executing guard value modifying instructions which are held in the main instruction queue and for updating said master guard value store with any modified guard value. Clearly, if you have predicate registers, then you need to have instructions which update predicate registers. Some of these instructions are shown on pages 7-19 and 7-41. The compare instructions are clearly not branch instructions, and therefore, since they are not branch instructions, they would not be held in the subsidiary queue. For instance, the compare instruction would be in the ALU's main instruction queue.

43. Referring to claim 3, Witt in view of Intel has taught a computer system as described in claim 2. Witt in view of Intel has not explicitly taught that the instruction supply circuitry comprises a control unit common to each parallel execution unit, said control unit including queue checking circuitry which is operable to check, before each sendguard instruction is supplied to the execution pipeline, that no earlier guard value modifying instructions affecting

the guard value requested by the sendguard instruction is still waiting in the main instruction queue. However, Official Notice is taken that stalling due to hazards caused by data dependencies is well known and accepted in the art. For instance, if a compare instruction in Intel immediately preceded a sendguard (branch) instruction which references the same predicate written by the preceding compare instruction, then the sendguard instruction will have to wait until the compare instruction is finished setting the predicate value. Otherwise, the sendguard instruction may be executed improperly. Therefore, in order to prevent incorrect execution, it would have been obvious to one of ordinary skill in the art at the time of the invention to have instruction supply circuitry that comprises a control unit common to each parallel execution unit, said control unit including queue checking circuitry which is operable to check, before each sendguard instruction is supplied to the execution pipeline, that no earlier guard value modifying instructions affecting the guard value requested by the sendguard instruction is still waiting in the main instruction queue.

44. Referring to claim 4, Witt in view of Intel has taught a computer system as described in claim 2. Witt in view of Intel has not explicitly taught that each execution unit comprises a plurality of pipelined stages with synchronized pipelined cycles for each of the execution units and the circuitry for executing guard value modifying instructions is located in a pipelined stage downstream of an earlier pipelined stage operable to execute sendguard instructions. However, Official Notice is taken that pipelining and its advantages are well known and accepted in the art. Pipelining allows for multiple instructions to be executed in parallel, thereby increasing instruction throughput and execution speed. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to pipeline each execution unit. Furthermore,

a person of ordinary skill in the art would have recognized that the execution of a modifying instruction would not be totally complete until the guard value is actually modified, where these updates (register writing) are well known in the art to occur in the write-back (final) stage of the pipeline. On the other hand, the sendguard instruction merely needs to read a guard value (predicate), and this would be done before execution, which inherently precedes the write back stage (you can't write a result until the execution completes). As a result, since writing a result to a destination occurs in a stage after (downstream) a stage in which an operand is read, the limitation of claim 4 is realized.

45. Referring to claim 5, Witt in view of Intel has taught a computer system as described in claim 4. Witt in view of Intel has not explicitly taught switching circuitry in the earlier pipelined stage for supplying a guard value which has been modified by a guard value modifying instruction in the subsequent pipelined stage in a machine cycle responsive to a sendguard instruction in the earlier pipelined stage in the same machine cycle. However, Official Notice is taken that result bypassing (forwarding) and its advantages are well known and accepted in the art. More specifically, if a result has already been calculated but not yet written to its final destination, and a dependent instruction needs the result to execute, then this result may be forwarded directly to the dependent instruction so that it doesn't have to wait for it to be written to the destination. This prevents the dependent instruction from stalling and once again results in increased throughput. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Witt in view of Intel to include switching circuitry in the earlier pipelined stage for supplying a guard value which has been modified by a guard value

modifying instruction in the subsequent pipelined stage in a machine cycle responsive to a sendguard instruction in the earlier pipelined stage in the same machine cycle.

46. Referring to claim 6, Witt in view of Intel has taught a computer system as described in claim 1. Witt in view of Intel has not explicitly taught that each execution unit comprises a plurality of pipelined stages with synchronized pipelined cycles for each of the execution units. However, Official Notice is taken that pipelining and its advantages are well known and accepted in the art. Pipelining allows for multiple instructions to be executed in parallel, thereby increasing instruction throughput and execution speed. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to pipeline each execution unit.

47. Referring to claim 7, Witt in view of Intel has taught a computer system as described in claim 1. Witt in view of Intel has not explicitly taught that each execution unit comprises first, second, and third pipelined stages wherein the first and second pipelined stages contain circuitry for executing sendguard instructions, and the third pipelined stage includes circuitry for executing guard value modifying instructions. However, Official Notice is taken that the general pipeline structure is well known and accepted in the art. More specifically, a pipeline is broken up into multiple stages which broadly include: a fetch stage, a decode stage, an execute stage, a memory stage, and a write-back stage. Operands are read (in decode) before execution and results are written after execution (in write back). Using this structure, sendguard instructions, which read guards (have guards sent to the unit executing the sendguard instruction) would read the guard before execution. Therefore, the decode and execute stages would be first and second stages which allow for execution of sendguard instructions while a write-back stage is the third

Art Unit: 2183

stage which allows for execution of guard modifying instruction (modified guards are written during write-back). Pipelining allows for multiple instructions to be executed in parallel, thereby increasing instruction throughput and execution speed. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to pipeline each execution unit.

48. Referring to claim 8, Witt in view of Intel has taught a computer system as described in claim 7. Furthermore, claim 8 is rejected for the same reasons set forth in the rejection of claim 5.

49. Referring to claim 9, Witt in view of Intel has taught a computer system as described in claim 1. Witt has further taught that:

a) said execution units include execution pipelines providing access to a data memory. See Fig. 1A-1B and note that the units access data memory and cache 55 and 70, respectively.

b) said pipelines including a first set of pipelines for use in executing instructions needed for memory access operations and a second set of pipelines arranged to carry out arithmetic operations, thereby providing decoupling of memory access operations from arithmetic operations. See Fig. 1A and note that the ALU and shifter (arithmetic) are separated from load and store units, thereby decoupling memory and arithmetic operations.

50. Referring to claim 10, Witt in view of Intel has taught a computer system as described in claim 9. Witt has further taught that two parallel pipelines are provided for arithmetic operations and access a common set of data registers including said master guard value store. See Fig. 1A and note that the shifter and ALU (two parallel units) access a common set of registers 30 and

Art Unit: 2183

they would each access the predicated register file taught by Intel because shift and ALU instructions in Intel are guarded. See pages 7-3 and 7-166, for instance.

51. Referring to claim 11, Witt in view of Intel has taught a computer system as described in claim 1. Intel has further taught that said master guard value store comprises a register file. See page 3-3 and 3-4.

52. Referring to claim 12, it has been noted that the computer system of claim 1 performs the method of claim 12. Therefore, claim 12 is rejected for the same reasons set forth in the rejection of claim 1.

53. Referring to claim 13, Terada in view of Mills and further in view of Irwin has taught a method as described in claim 12. Furthermore, it has been noted that the computer system of claim 2 performs the method of claim 13. Therefore, claim 13 is rejected for the same reasons set forth in the rejection of claim 2.

54. Referring to claim 14, Terada in view of Mills and further in view of Irwin has taught a method as described in claim 13. Furthermore, it has been noted that the computer system of claim 3 performs the method of claim 14. Therefore, claim 14 is rejected for the same reasons set forth in the rejection of claim 3.

55. Claims 15-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Intel, as applied above, in view of Hennessy and Patterson, Computer Architecture - A Quantitative Approach, 2<sup>nd</sup> Edition, 1996 (as disclosed by applicant and herein referred to as Hennessy).

56. Referring to claim 15, Intel has taught a pipelined execution unit for a computer system which comprises at least two pipelined stages (it is inherent that the Intel IA-64 architecture

Art Unit: 2183

involves a pipeline unit having at least two stages), with an earlier one of the pipelined stages including sendguard circuitry responsive to a sendguard instruction to dispatch a guard value for a guard indicator defined in the sendguard instruction, and a later one of the pipelined stages including guard value modifying circuitry for executing guard value modifying instruction which cause the value of a guard indicator to be modified (results are written, by guard-modifying instructions, such as the compare instruction shown on page 7-19, at the end of a pipeline, while guard values are read earlier, by sendguard instructions such as the add instruction shown on page 7-3, because execution depends on its value, and therefore it must be known before execution occurs), the pipelined execution unit further comprising:

- a) a master guard value store for holding a master representation of current values for guard indicators. See page 3-3 and note the predicate register file.
- b) Intel has not explicitly taught means for determining any dependencies between a sendguard instruction in the earlier pipelined stage and a guard modifying instruction in the later pipelined stage and switching circuitry for selecting when a sendguard instruction is executed and responsive to any such dependencies, whether a guard value held in the master guard value store or a guard value just modified by the guard value modifying circuitry is to be dispatched.

However, Hennessy has taught the concept of forwarding wherein dependences are detected between two instructions and data that has been produced by a first instruction but not written yet is supplied directly to the second (dependent) instruction in order to prevent delays. See pages 147-150. A person of ordinary skill in the art would have recognized that dependencies must be checked in order to ensure that the correct data is being used by all dependent instructions and that forwarding is a good technique, which allows for the elimination of unnecessary pipeline

Art Unit: 2183

stalls, thereby increasing the speed of execution. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to check for dependencies and supply the correct guard value depending on any detected dependency.

57. Referring to claim 16, Intel in view of Hennessy has taught a pipelined execution unit as described in claim 15. Furthermore, claim 16 is rejected for the same reasons set forth in the rejection of claim 7 above.

58. Referring to claim 17, Intel in view of Hennessy has taught a pipelined execution unit as described in claim 15. Furthermore, it is inherent that a data write-back stage exists for writing back the results of execution of an instruction into a data store. For instance, looking at page 7-3, when the add instruction finishes executing, a result will be written into the destination register in the write-back stage.

59. Referring to claim 18, Intel has taught a method of executing instructions in a pipelined execution unit (it is inherent that the Intel IA-64 architecture involves a pipeline unit), each instruction having a respective guard indicator (note the reference to predicate (qp) in each of the instructions) selected from a set of guard indicators (page 3-3, predicate register file) and execution of the instructions being predicated on values of the guard indicators (see page 3-4), wherein the instructions include a sendguard instruction which, when executed, causes transfer of a guard value from the pipelined execution unit (see page 7-127) and a guard value modifying instruction which modifies the value of a guard indicator (see page 7-19), the method comprising:

Art Unit: 2183

a) supplying instructions to the pipelined execution unit including said sendguard instructions and said guard value modifying instructions. It is inherent that all instructions must be supplied to an execution unit; otherwise they cannot be executed.

b) Intel has not explicitly taught checking dependencies between guard value modifying instructions supplied to the pipelined execution unit earlier than a sendguard instruction relating to a same guard indicator and supplying the guard value of the guard indicator requested in the sendguard instruction selectively from a master guard value store or guard value modifying circuitry in dependence on the results of said dependency checks, so as to ensure that the guard value of a guard indicator which is dispatched responsive to a sendguard instruction is correct in relation to any earlier guard value modifying instructions in the pipelined execution unit.

However, Hennessy has taught the concept of forwarding wherein dependences are detected between two instructions and data that has been produced by a first instruction but not written yet is supplied directly to the second (dependent) instruction in order to prevent delays. See pages 147-150. A person of ordinary skill in the art would have recognized that dependencies must be checked in order to ensure that the correct data is being used by all dependent instructions and that forwarding is a good technique, which allows for the elimination of unnecessary pipeline stalls, thereby increasing the speed of execution. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to check for dependencies and supply the correct guard value depending on any detected dependency.

60. Referring to claim 19, it has been noted that the execution unit of claim 15 operates in the same way as the execution unit of claim 19. Therefore, claim 19 is rejected for the same reasons set forth in the rejection of claim 15.

61. Claims 1-6 and 8-14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Terada et al., U.S. Patent No. 6,041,399 (as disclosed by applicant and herein referred to as Terada) in view of Mills, U.S. Patent No. 5,889,984 (as disclosed by applicant), and further in view of Irwin, U.S. Patent No. 6,393,026 B1 (as disclosed by applicant).

62. Referring to claim 1, Terada has taught a computer system for executing instructions having assigned guard indicators, the system comprising:

- a) instruction supply circuitry and a plurality of parallel execution units for receiving respective instructions from the supply circuitry. See Fig.5 and note the multiple processing units 1 and 2, and the supply circuitry from the instruction cache.
- b) each instruction having a respective guard indicator selected from a set of guard indicators, common to the plurality of execution units. See Fig.5 and column 5, lines 20-23. Note that the "P" field is used to select a predicate register from a set of predicate registers.
- c) one of said execution units including a master guard value store containing a master representation of current values for the guard indicators in said set of guard indicators and guard value transfer circuitry operable to transfer a guard value from said master store to another of said execution units in response to a sendguard instruction being executed in said one execution unit. See Fig.5, Fig.9, and column 7, lines 1-17. Note that Fig.5 shows the coupling of the predicate register files so that values can be transferred. In addition, Terada's system can be interpreted such that the master guard value store is located in the execution unit in which a sendguard instruction is executed. For instance, from Fig.9, the "cmp.gt r5,#9,p0,B" instruction will be executed in execution unit 2 (in Fig.5). Therefore, the master guard store would be

Art Unit: 2183

interpreted to be in execution unit 2. The aforementioned "cmp.gt" instruction is a combined predicate-modification and sendguard instruction. Its function is to modify a predicate and optionally broadcast the predicate so that it can also be written to the predicate register file of execution unit 1 (or any other execution unit in the system). Terada has not explicitly taught an instruction that only sends the guard values. However, Mills has taught a concept in which separate instructions are used to modify a predicate register and transfer the modified predicate to another functional unit. For instance, see Fig.6 and column 6, line 63, to column 7, line 35. A person of ordinary skill in the art would have recognized that by splitting up the combined modify-send instruction of Terada into two individual instructions as in Mills, certain advantages would be realized. The first advantage would be increased flexibility for the programmer. In the current system, either all of the predicate registers (in each of the units) will be updated or no additional predicate registers (in addition to the master store) will be updated. This may be undesired by the programmer in cases where functional unit A is executing instructions based on a local set of predicates. However, in the current system if another functional unit B needs to be updated after an update to master store in functional unit C, then the local predicates of functional unit A must also be updated. On the contrary, by splitting up the instructions, the programmer is allowed to selectively choose which guard registers are updated for which units, providing more flexibility, and in essence, a greater number of total predicates by having both local and global predicates. This would be done with the instruction format shown by Mills in Fig4A and Fig.4B, where the format includes the source unit and the destination unit along with what data will be transferred. Overall, time and work required by the processor would also be reduced by not having to update unnecessary guard registers. Therefore, it would have been

Art Unit: 2183

obvious to one of ordinary skill in the art at the time of the invention to replace the combined guard-modify-send instruction with separate instructions (a guard-modify instruction and a sendguard instruction) as taught by Mills.

d) Terada has not explicitly taught that the instruction supply circuitry comprises a main instruction queue for holding instructions to be supplied to the parallel execution units and a subsidiary instruction queue for holding sendguard instructions, with the subsidiary queue having priority access to the execution pipelines to avoid unnecessary delays for execution of sendguard instructions. However, Official Notice is taken that the concept of data coherency is well known and expected in the art. Coherency is the idea of updating multiple memories so that the contents of each of the memories are identical. Through this updating, each memory will contain the most recent and valid data. It should be noted that coherency is achieved through the execution of a sendguard instruction as mentioned above in part (c). Where coherency is desired, it is imperative, especially for parallel systems, that an instruction set to execute in functional unit B that uses a guard that has just been modified by functional unit A, will use the new guard value (after modification) instead of the previous guard value (before modification). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to assign the sendguard instruction highest priority. By doing this, the guard registers in each of the functional units that require coherency will be updated before any other instruction accesses the guard register. This ensures that all predicated instructions would have the most recent predicates. With this concept in mind, it should be noted that Irwin has taught the general concept of a system that has two instruction queues. The first instruction queue contains all high-priority tasks that are time critical. The second queue contains all other tasks (of lower

Art Unit: 2183

priority) that are not time critical. Items in the first queue are always processed before items in the second queue. A person of ordinary skill in the art would have recognized that such a system would be directly applicable to the system of Terada in view of Mills. The first queue would be used to store sendguard instructions, which have high priority and are to be executed in order to avoid delays. The second queue would be used to hold the rest of the instructions that will be executed when they are not preempted due to sendguard instructions. In summation, it would have been obvious to one of ordinary skill in the art at the time of the invention to include a subsidiary high-priority queue to hold sendguard instructions and a main lower-priority instruction queue to hold non-sendguard instructions.

63. Referring to claim 2, Terada in view of Mills and further in view of Irwin has taught a computer system as described in claim 1. Terada has further taught that said execution unit includes circuitry for executing guard value modifying instructions which are held in the main instruction queue and for updating said master guard value store with any modified guard value. See Fig.8 and note that the "cmp.gt" instructions are used to modify the guard values. According to the rejection of claim 1, since the main queue holds everything but sendguard instructions, then this modify instruction would be located in the main instruction queue.

64. Referring to claim 3, Terada in view of Mills and further in view of Irwin has taught a computer system as described in claim 2. Furthermore, a person of ordinary skill in the art would have recognized that checking for dependencies between instructions is well known in the art. For instance, suppose the following code was encountered:

Cmp.gt p0,r1,r2	//set predicate p0 if r1>r2 (in the integer unit)
I2F p0	//send predicate p0 from the integer unit to the float unit

Art Unit: 2183

According to the system taught by Terada in view of Mills in view of Irwin, the Cmp.gt instruction will be put in the main queue and the I2F instruction will be put in the subsidiary queue. Since the subsidiary queue holds sendguard instructions that are high priority, the I2F instruction would be executed before the Cmp.gt instruction if dependencies were not checked. If this were the case, then predicate p0 may not be the correct value when it is being sent by the I2F instruction since it would not have been previously modified by the Cmp.gt instruction. This would cause inconsistent and incorrect results. Therefore, in order to make sure the predicate being sent reflects the value calculated by an instruction that has not yet executed but appears before the sendguard instruction, it would have been obvious to one of ordinary skill in the art at the time of the invention to have instruction supply circuitry that comprises a control unit common to each parallel execution unit, said control unit including queue checking circuitry which is operable to check, before each sendguard instruction is supplied to the execution pipeline, that no earlier guard value modifying instructions affecting the guard value requested by the sendguard instruction is still waiting in the main instruction queue.

65. Referring to claim 4, Terada in view of Mills and further in view of Irwin has taught a computer system as described in claim 2. Terada has further taught that each execution unit comprises a plurality of pipelined stages with synchronized pipelined cycles for each of the execution units. See the first 3 lines of the abstract and note that by stating that the processing units are of the pipelined type, the processing units include multiple stages for processing an instruction. Terada has not explicitly taught that the circuitry for executing guard value modifying instructions is located in a pipelined stage downstream of an earlier pipelined stage operable to execute sendguard instructions. However, a person of ordinary skill in the art would

have recognized that the execution of a modifying instruction would not be totally complete until the guard value is actually modified, where these updates (register writing) are well known in the art to occur in the write-back (final) stage of the pipeline. On the other hand, the sendguard instruction merely needs to pass along a value, and this would be done as early as the execution stage. In addition, it would have been realized by one of ordinary skill in the art that the sendguard instructions should be executed as soon as possible in order to quickly update predicate registers throughout each of the processing units that require these updates. Therefore, in order to increase efficiency, it would have been obvious to one of ordinary skill in the art at the time of the invention to have circuitry for executing guard value modifying instructions located in a pipelined stage downstream of an earlier pipelined stage operable to execute sendguard instructions.

66. Referring to claim 5, Terada in view of Mills and further in view of Irwin has taught a computer system as described in claim 4. Terada has further taught switching circuitry in the earlier pipelined stage for supplying a guard value which has been modified by a guard value modifying instruction in the subsequent pipelined stage in a machine cycle responsive to a sendguard instruction in the earlier pipelined stage in the same machine cycle. See Fig.5, components 106 and 206, and Fig.13, components 11 and 12. These components supply guard values that have been modified by guard-modifying instructions in response to a sendguard instruction.

67. Referring to claim 6, Terada in view of Mills and further in view of Irwin has taught a computer system as described in claim 1. Terada has further taught that each execution unit comprises a plurality of pipelined stages with synchronized pipelined cycles for each of the

Art Unit: 2183

execution units. See the first 3 lines of the abstract and note that the processing units are pipelined and they perform parallel operations. Also, from Fig.8 and column 6, lines 25-61, it can be seen how each processing unit displays parallelism.

68. Referring to claim 8, Terada in view of Mills and further in view of Irwin has taught a computer system as described in claim 1. Terada has further taught switching circuitry for selecting, responsive to a sendguard instruction, whether a guard value held in the master guard value store or a guard value just modified by a guard value modifying instruction is dispatched via the guard value transfer circuitry. See Fig.5, components 106 and 206, and column 7, lines 18-33.

69. Referring to claim 9, Terada in view of Mills and further in view of Irwin has taught a computer system as described in claim 1. Terada has further taught:

a) said execution units include execution pipelines providing access to a data memory. See Fig.5 and note that the data cache 4 is coupled to the execution units. Also, in Fig.16, it can be seen that the execution units are connected to memory 23.

b) said pipelines including a first set of pipelines for use in executing instructions needed for memory access operations and a second set of pipelines arranged to carry out arithmetic operations, thereby providing decoupling of memory access operations from arithmetic operations. From Fig.12 it can be seen that a second pipeline carries out arithmetic operations by performing the "cmp.eq" instructions shown in the first box. Also, from Fig.12, a first pipeline carries out memory access operations by performing the "ld" instructions shown in the second box.

70. Referring to claim 10, Terada in view of Mills and further in view of Irwin has taught a computer system as described in claim 9. Terada has further taught that two parallel pipelines are provided for arithmetic operations and access a common set of data registers including said master guard value store. See Fig.14 and column 9, lines 40-51. Note that two ALUs 103 and 104 access a common data register file and also, from the aforementioned passage, the instructions executed by both of the ALUs can access a common "master" guard value store by broadcasting a read value.

71. Referring to claim 11, Terada in view of Mills and further in view of Irwin has taught a computer system as described in claim 1. Terada has further taught that said master guard value store comprises a register file. See column 5, lines 52-54, and note that the master guard value store is a register file with a plurality of regions (entries). Recall that the master guard value store is considered to be in the execution unit that is currently executing a sendguard instruction.

72. Referring to claim 12, it has been noted that the computer system of claim 1 performs the method of claim 12. Therefore, claim 12 is rejected for the same reasons set forth in the rejection of claim 1.

73. Referring to claim 13, Terada in view of Mills and further in view of Irwin has taught a method as described in claim 12. Furthermore, it has been noted that the computer system of claim 2 performs the method of claim 13. Therefore, claim 13 is rejected for the same reasons set forth in the rejection of claim 2.

74. Referring to claim 14, Terada in view of Mills and further in view of Irwin has taught a method as described in claim 13. Furthermore, it has been noted that the computer system of

Art Unit: 2183

claim 3 performs the method of claim 14. Therefore, claim 14 is rejected for the same reasons set forth in the rejection of claim 3.

75. Claim 7 is rejected under 35 U.S.C. 103(a) as being unpatentable over Terada in view of Mills in view of Irwin, as applied above, and further in view of Palanca et al., U.S. Patent No. 6,216,215 B1 (as disclosed by applicant and herein referred to as Palanca).

76. Referring to claim 7, Terada in view of Mills and further in view of Irwin has taught a computer system as described in claim 1. Terada in view of Mills and further in view of Irwin has not explicitly taught that each execution unit comprises first, second, and third pipelined stages wherein the first and second pipelined stages contain circuitry for executing sendguard instructions, and the third pipelined stage includes circuitry for executing guard value modifying instructions. In general, Terada in view of Mills and further in view of Irwin has not taught that the sendguard instruction is executed and retired early. However, Palanca has taught the concept of executing and retiring certain types of instructions early for purposes of reducing pipeline delay. See column 6, line 66, to column 7, line 10. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to execute and retire sendguard instructions early in the system of Terada in view of Mills and further in view of Irwin as taught by Palanca because Palanca has shown that such a concept leads to the reduction of pipeline delay.

Art Unit: 2183

77. Claims 15 and 17-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Terada in view of Mills in view of Irwin, as applied above, and further in view of Hennessy, as applied above.

78. Referring to claim 15, it has been noted that the computer system of claim 5 operates in the same fashion as the pipelined execution unit for a computer system described in the first paragraphs (lines 1-10) of claim 15. Therefore, this portion of claim 15 is rejected for the same reasons set forth in the rejection of claim 5.

a) Terada has further taught a master guard value store for holding a master representation of current values for guard indicators. Terada's system can be interpreted such that the master guard value store is located in the execution unit in which a sendguard instruction is executed. For instance, from Fig.9, the "cmp.gt r5,#9,p0,B" instruction will be executed in execution unit 2 (in Fig.5). Therefore, the master guard store would be interpreted to be in execution unit 2.

b) Terada in view of Mills and further in view of Irwin has not explicitly taught a dependency determiner for determining any dependencies between a sendguard instruction in the earlier pipelined stage and a guard modifying instruction in the later pipelined stage and switching circuitry for selecting when a send guard instruction is executed and responsive to any such dependencies, whether a guard value held in the master guard value store or a guard value just modified by the guard value modifying circuitry is to be dispatched. However, Hennessy has taught the concept of forwarding wherein dependences are detected between two instructions and data that has been produced by a first instruction but not written yet is supplied directly to the second (dependent) instruction in order to prevent delays. See pages 147-150. A person of ordinary skill in the art would have recognized that dependencies must be checked in order to

Art Unit: 2183

ensure that the correct data is being used by all dependent instructions and that forwarding is a good technique, which allows for the elimination of unnecessary pipeline stalls, thereby increasing the speed of execution. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to check for dependencies and supply the correct guard value depending on any detected dependency.

79. Referring to claim 17, Terada in view of Mills in view of Irwin and further in view of Hennessy has taught a pipelined execution unit as described in claim 15. Terada has further taught that his system is pipelined. See the first 3 lines of the abstract. Official Notice is taken that write-back stages are well known and expected stages in the art of pipelining. From Fig.5, it can be seen that an output of each ALU is coupled to the ALU's respective data register file so that writes can be performed. Since write-back stages are well known in the art and a write to the data register would occur after execution, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a data write-back stage for writing back the results of execution of an instruction into a data store.

80. Referring to claim 18, it has been noted that the computer system of claim 1 performs the method described in the first two paragraphs (lines 1-12) of claim 18. Therefore, this portion of claim 18 is rejected for the same reasons set forth in the rejection of claim 1. Terada in view of Mills and further in view of Irwin has not explicitly taught checking dependencies between guard value modifying instructions supplied to the pipelined execution unit earlier than a sendguard instruction relating to the same guard indicator and supplying the guard value of the guard indicator requested in the sendguard instruction selectively from a master guard value store or guard value modifying circuitry in dependence on the results of said dependency checks, so as to

Art Unit: 2183

ensure that the guard value of a guard indicator which is dispatched responsive to a sendguard instruction is correct in relation to any earlier guard value modifying instructions in the pipelined execution unit. However, Hennessy has taught the concept of forwarding wherein dependences are detected between two instructions and data that has been produced by a first instruction but not written yet is supplied directly to the second (dependent) instruction in order to prevent delays. See pages 147-150. A person of ordinary skill in the art would have recognized that dependencies must be checked in order to ensure that the correct data is being used by all dependent instructions and that forwarding is a good technique, which allows for the elimination of unnecessary pipeline stalls, thereby increasing the speed of execution. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to check for dependencies and supply the correct guard value depending on any detected dependency.

81. Referring to claim 19, it has been noted that the computer system of claim 5 operates in the same fashion as the pipelined execution unit for a computer system described in the first paragraphs (lines 1-10) of claim 19. Therefore, this portion of claim 19 is rejected for the same reasons set forth in the rejection of claim 5.

a) Terada has further taught a master guard value store for holding a master representation of current values for guard indicators. Terada's system can be interpreted such that the master guard value store is located in the execution unit in which a sendguard instruction is executed.

For instance, from Fig.9, the "cmp.gt r5,#9,p0,B" instruction will be executed in execution unit 2 (in Fig.5). Therefore, the master guard store would be interpreted to be in execution unit 2.

b) Terada in view of Mills and further in view of Irwin has not explicitly taught a dependency determiner for determining any dependencies between a sendguard instruction in the earlier

Art Unit: 2183

pipelined stage and a guard modifying instruction in the later pipelined stage and switching circuitry for selecting when a send guard instruction is executed and responsive to any such dependencies, whether a guard value held in the master guard value store or a guard value just modified by the guard value modifying circuitry is to be dispatched. However, Hennessy has taught the concept of forwarding wherein dependences are detected between two instructions and data that has been produced by a first instruction but not written yet is supplied directly to the second (dependent) instruction in order to prevent delays. See pages 147-150. A person of ordinary skill in the art would have recognized that dependencies must be checked in order to ensure that the correct data is being used by all dependent instructions and that forwarding is a good technique, which allows for the elimination of unnecessary pipeline stalls, thereby increasing the speed of execution. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to check for dependencies and supply the correct guard value depending on any detected dependency.

82. Claim 16 is rejected under 35 U.S.C. 103(a) as being unpatentable over Terada in view of Mills in view of Irwin in view of Hennessy, as applied above, and further in view of Palanca, as applied above

83. Referring to claim 16, Terada in view of Mills in view of Irwin and further in view of Hennessy has taught a pipelined execution unit as described in claim 15. Terada in view of Mills in view of Irwin and further in view of Hennessy has not explicitly taught that each execution unit comprises first, second, and third pipelined stages wherein the first and second pipelined stages contain circuitry for executing sendguard instructions, and the third pipelined stage

Art Unit: 2183

includes circuitry for executing guard value modifying instructions. In general, it has not been taught that the sendguard instruction is executed and retired early. However, Palanca has taught the concept of executing and retiring certain types of instructions early for purposes of reducing pipeline delay. See column 6, line 66, to column 7, line 10. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to execute and retire sendguard instructions early in the system of Terada in view of Mills in view of Irwin and further in view of Hennessy as taught by Palanca because Palanca has shown that such a concept leads to the reduction of pipeline delay.

### *Conclusion*

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (703) 305-7811. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703) 305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Art Unit: 2183

DJH

David J. Huisman

July 1, 2004



EDDIE CHAN  
SUPERVISORY PATENT EXAMINER  
TECHNOLOGY CENTER 2100